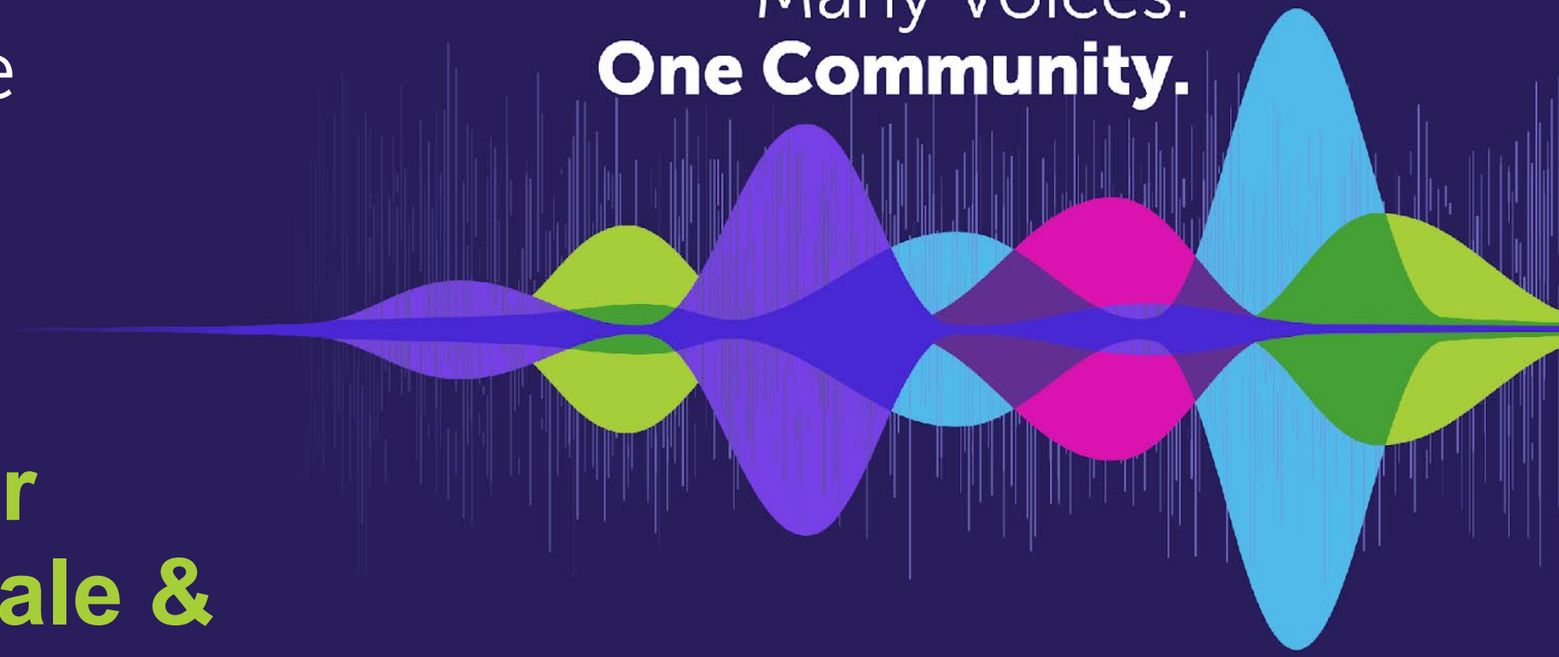
RSAC | 2025 Conference

Many Voices.
One Community.

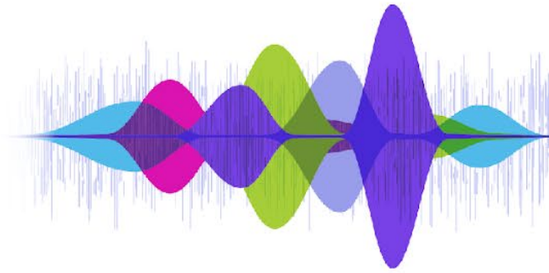


SESSION ID: APP-T09

Strategizing SAST for Speed, Accuracy, Scale & Dev Experience

Sandeep Jayashankar
Cybersecurity Architect
Product Security
SoFi

Pritam Mungse
Senior Director
Product Security
SoFi



Disclaimer

Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the presenters individually and, unless expressly stated to the contrary, are not the opinion or position of RSA Conference LLC or any other co-sponsors. RSA Conference LLC does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented.

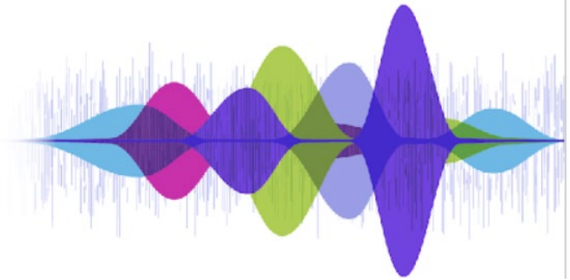
Attendees should note that sessions may be audio- or video-recorded and may be published in various media, including print, audio and video formats without further notice. The presentation template and any media capture are subject to copyright protection.

© 2025 RSA Conference LLC or its affiliates. The RSAC and RSAC CONFERENCE logos and other trademarks are proprietary. All rights reserved.

This content is provided for informational and educational purposes only and should not be construed as financial advice. No brands or products mentioned are affiliated with SoFi, nor do they endorse or sponsor this presentation. Third-party trademarks referenced herein are the property of their respective owners.

© 2025 Social Finance, Inc. All rights reserved

Purpose

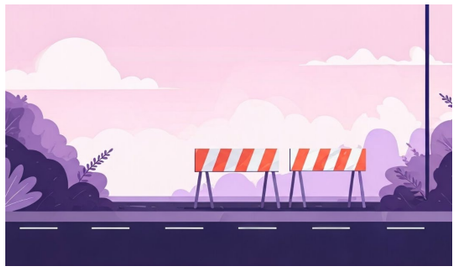


Share our Journey of SAST implementation at SoFi

with real-world implementation insights and lessons

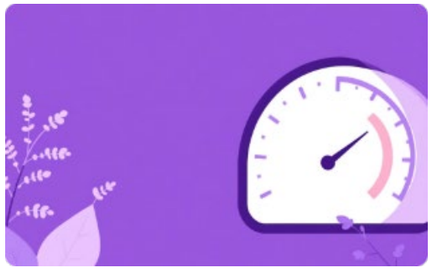
Our Agenda

Define Goals



Identify Constraints

Develop Strategy

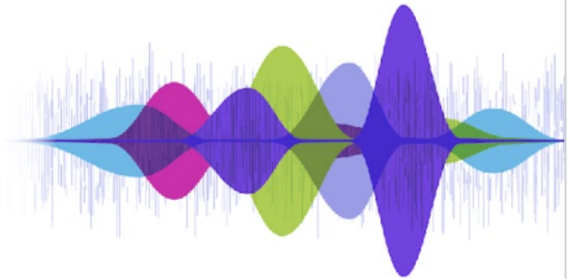


Measure Success

What's next

Key Takeaways

Real-World Case Studies



Org X: Compute-Rich

Setup: Homegrown orchestration/correlation with AI/ML categorization.

Challenges: High tech debt and operational concerns.

Org Y: Service-Oriented

Setup: Pipeline scanning with custom policies, dashboards.

Challenges: High false positives and inefficient correlation.

Org Z: Engineering-Centric

Setup: Open source scanners with custom vuln correlation.

Challenges: Inconsistent results, and lack of remediation.

Define Goals

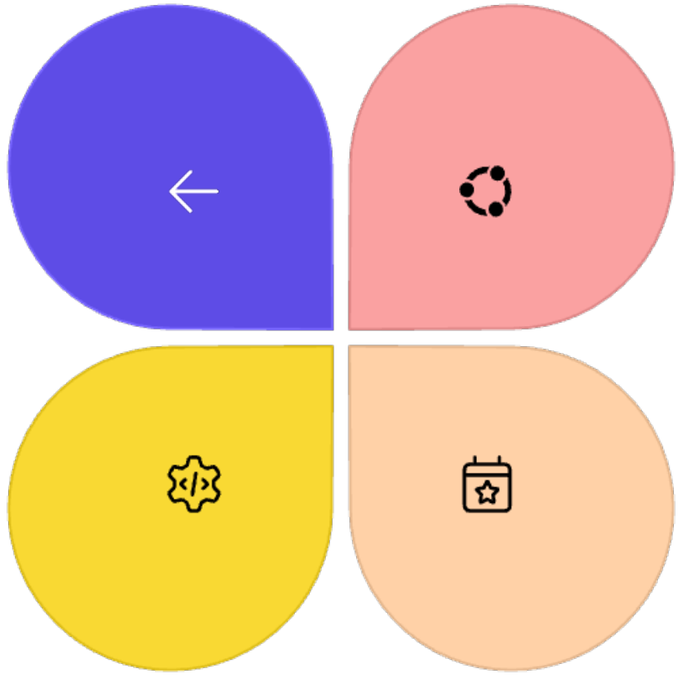
Define Goals

Shift Left

Catch issues before its costly

Enhance Adoption

Meet Developers where they code

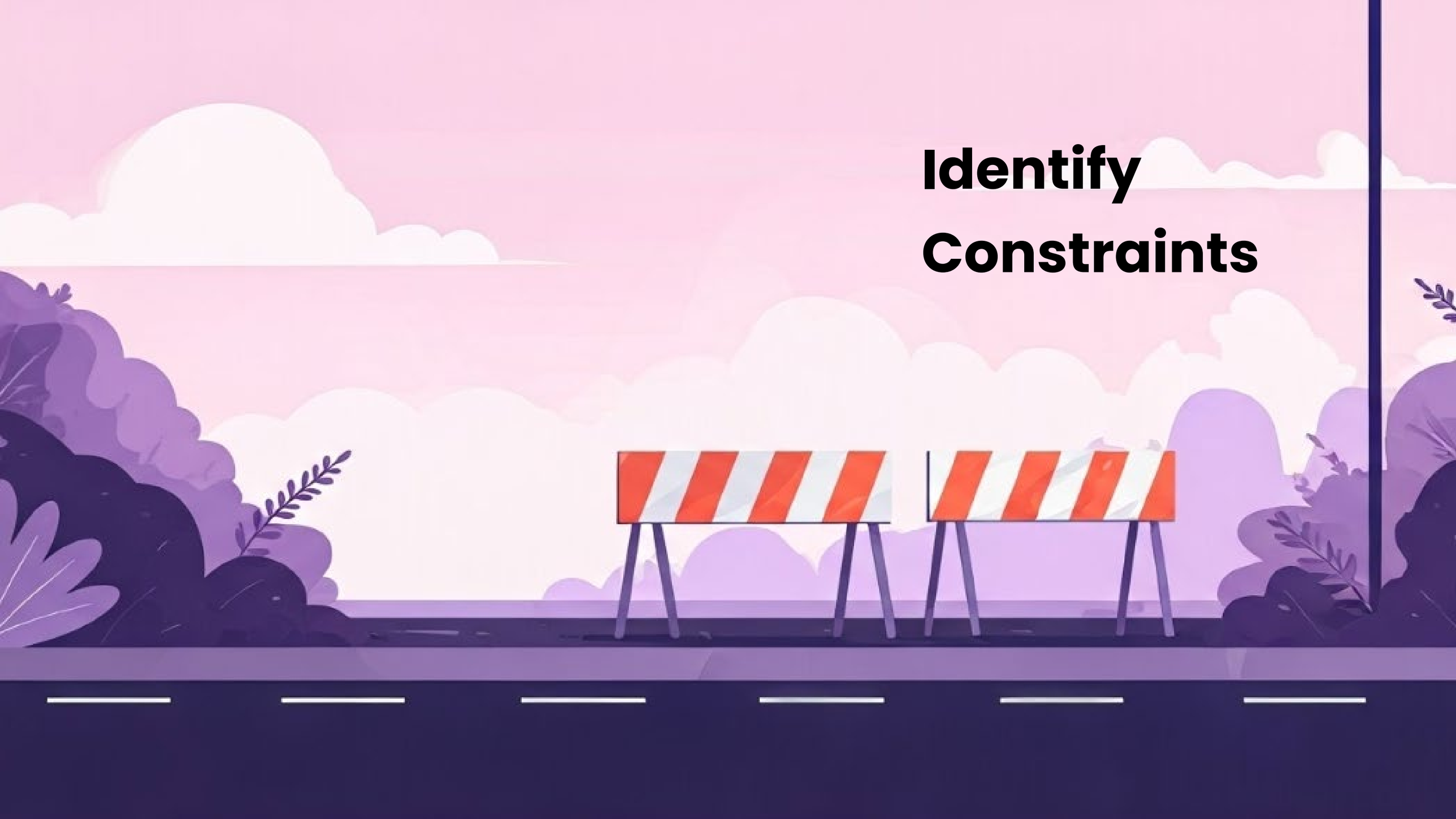


Operationalize

Streamlined scanning and enforcement

Prioritize

Context-Based Triaging with actionable remediation


Identify
Constraints

Identify Constraints

False Positives/Negatives

Consumes significant resources for manual investigation and verification

Actionable Remediation

Need to prioritize findings that developers can effectively address

Missing Context

Security tools often lack business risk and environmental context

Speed vs Depth

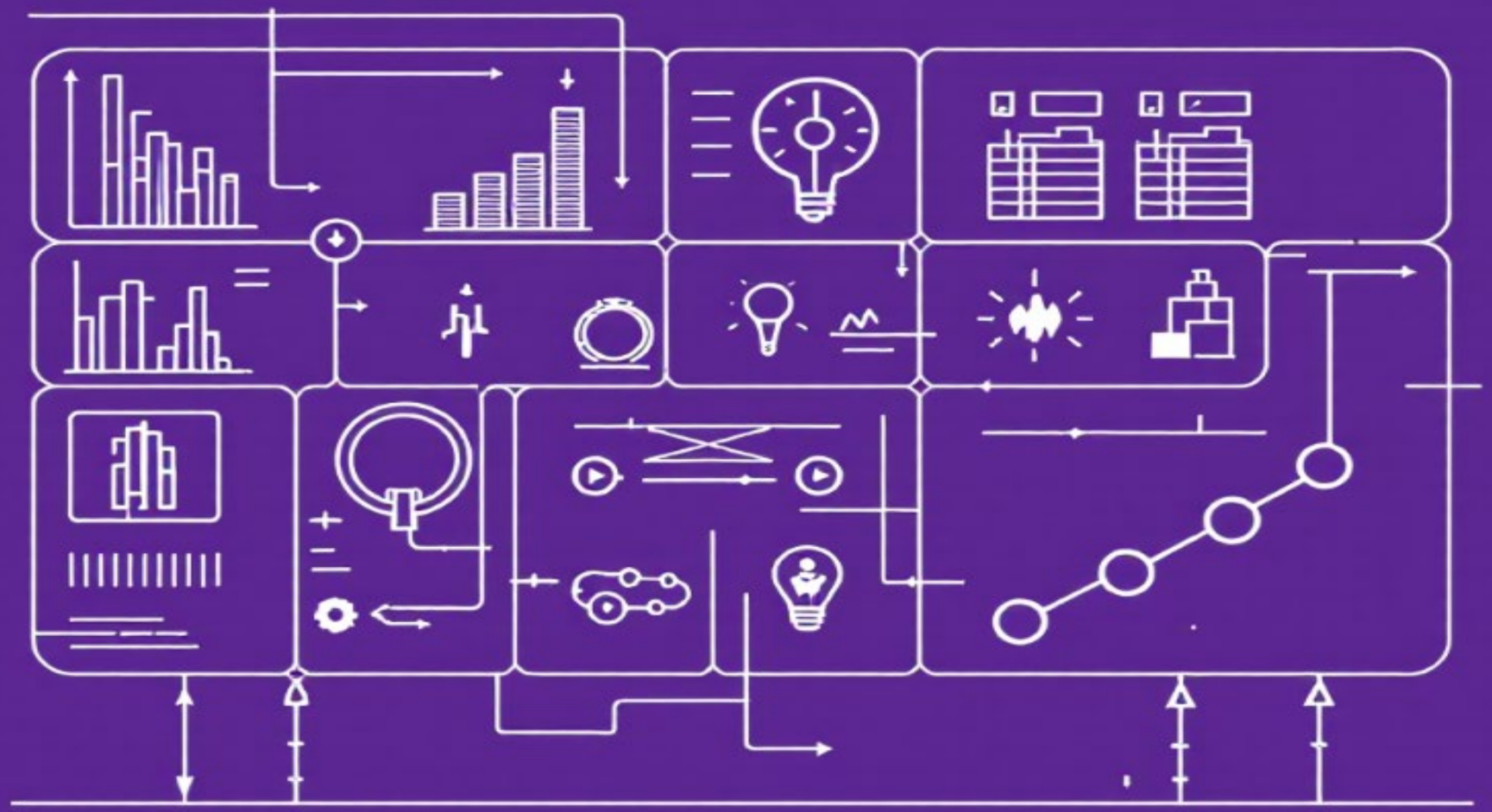
Challenging balance between thorough analysis and performance needs

Manage Rules

Complexity in maintaining and customizing scanning rule configurations

Dev Environment Integration

Inconsistent scanning processes are hindering productivity



Develop Strategy

Develop Strategy

ST1 - Evaluate/Benchmark Scanners

Standardize tooling evaluation based on company requirements.

ST2 - Define Scan Parameters

Define scan configurations for operationalizing SAST.

ST3 - Define Secure Build Pipeline

Apply Scan Parameters to our Build Pipeline

ST4 - Develop Contextual Risk Engine

Enable actionable remediation based on risk context.

ST5 - Implement End to End Orchestration

Seamless orchestration from dev to remediation

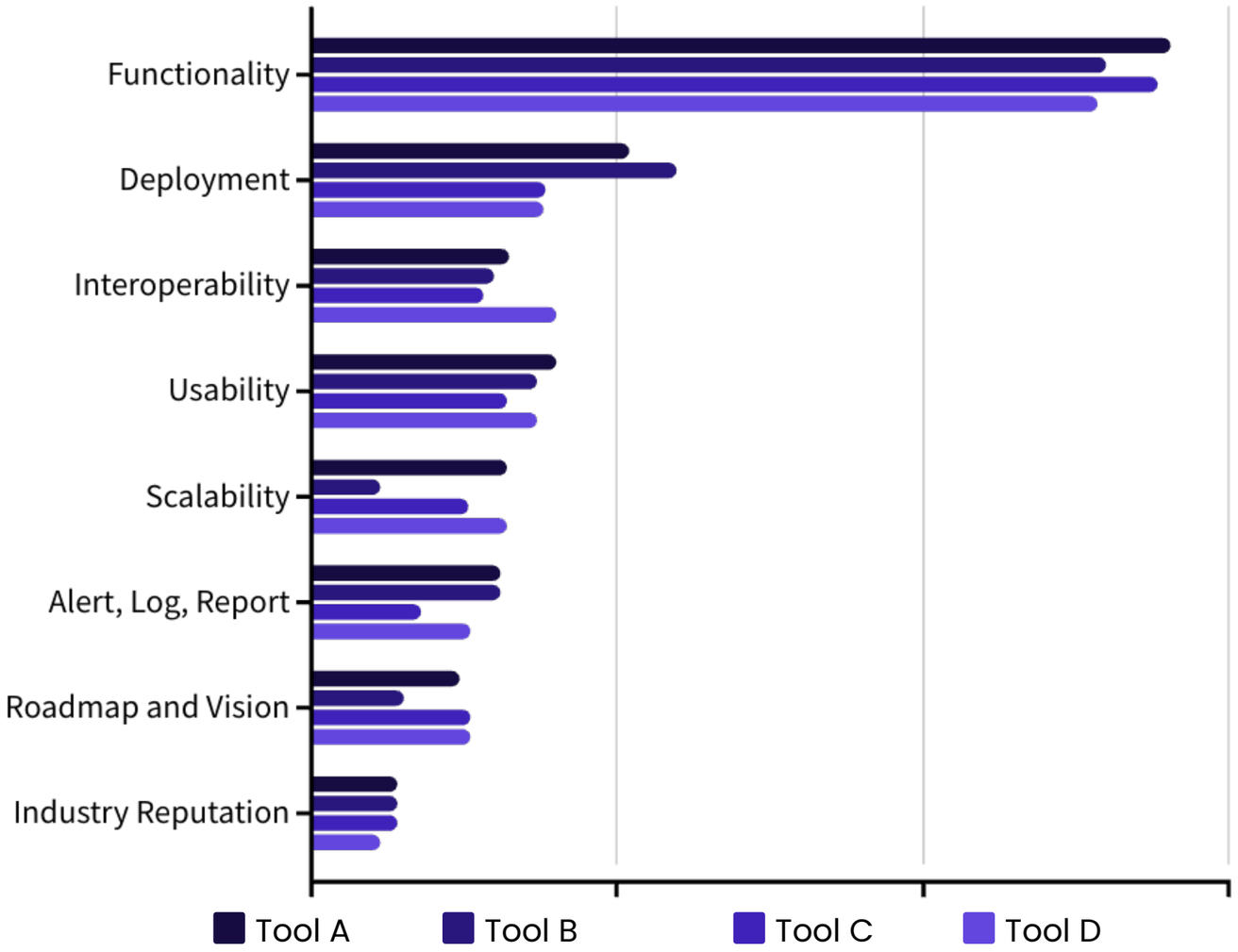
ST6 - Enhance Developer Adoption

Empower developers for improved security practices.

Build SAST Landscape

Step by Step coverage from basic scanners to full-blown SAST landscape

ST1 – Evaluate/Benchmark Scanners



Functionality Isn't Everything

Most tools have all the functionalities we need

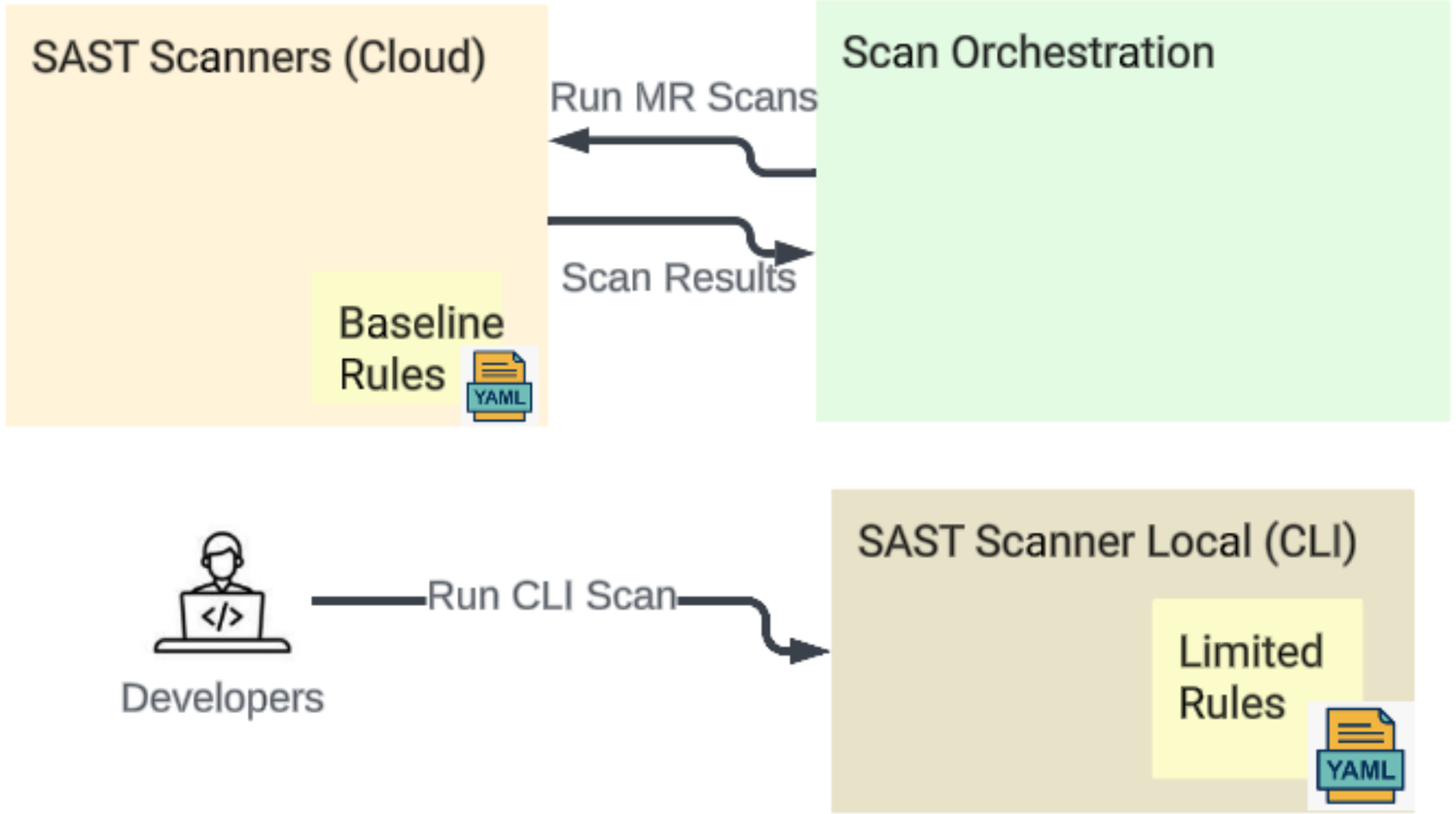
Operational Fit Matters

Deployment simplicity, Integrations with existing tools, and aligned technologies are critical factors as well.

Think Long-Term

Roadmap, vision, and scalability determine whether a tool can evolve with your ecosystem.

ST2 - Define Scan Parameters



Scan Orchestration

Speed vs Depth Tradeoffs

Baseline vs Custom Rules

Use-Case Driven

SAST Scanners

Scanners: lightweight, heavyweight

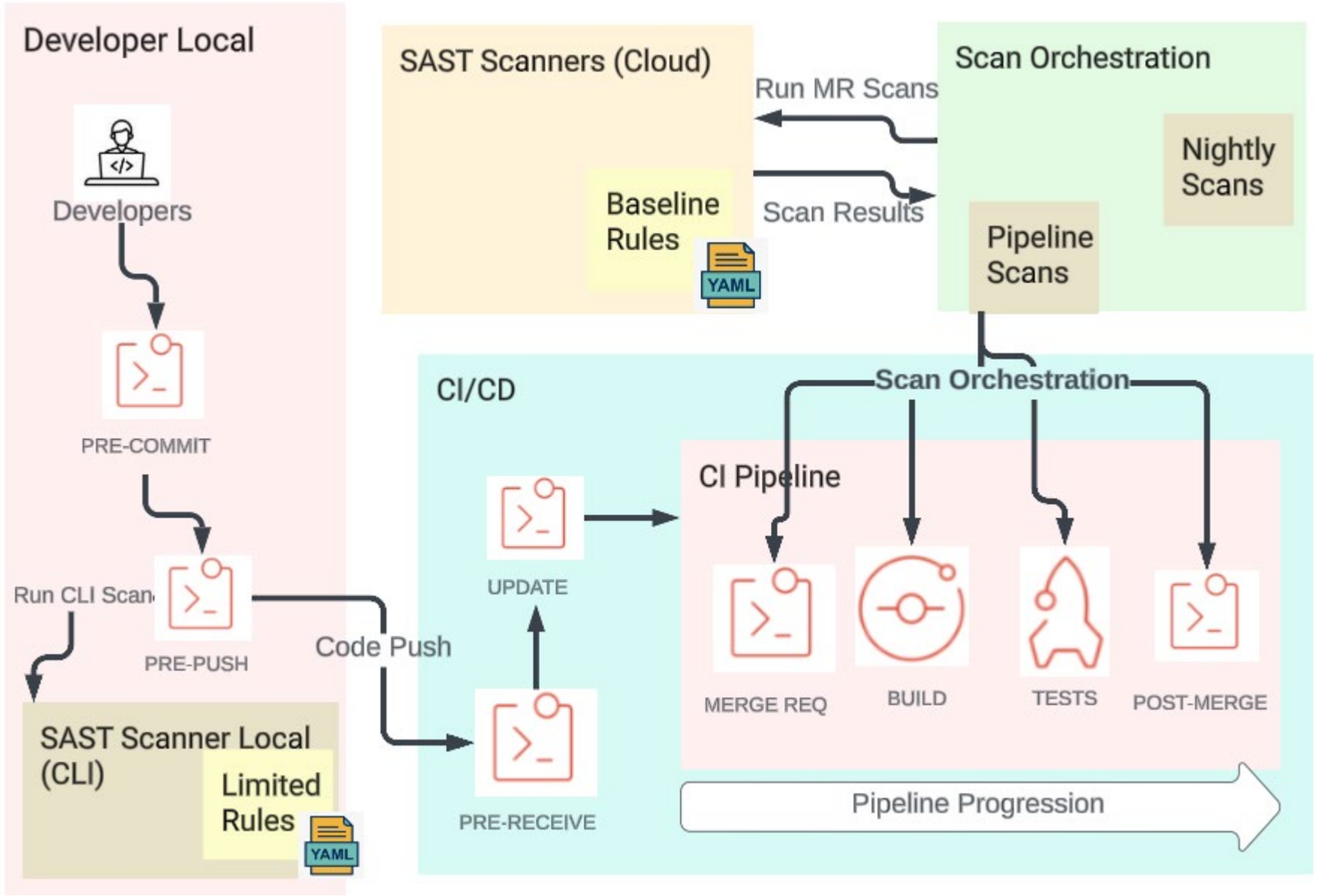
Scans: incremental, full, comprehensive scans

modes: local, cloud

ST2 – Define Scan Parameters

⚡ Incremental Scan	🔍 Full Scan	🧠 Comprehensive Scan
🎯 New code validation	🎯 Quick baseline check	🔬 Deep control/data flow
⏱️ Time: ~30 sec	⏱️ Time: ~1.5 min	⏱️ Time: ~45 min
Lightweight Scanners	Lightweight Scanners	Heavyweight Scanners
📜 Baseline + custom rules	📜 Baseline rules	📜 Baseline + extensive

ST3 – Define Secure Build Pipeline



Scan Orchestration

Define clear triggers and schedules for each scan tier.

Build Pipeline

Right enforcements at the right pipeline gates

Progressive Scan Depth

Increase scan thoroughness outside critical wait points.

SoFi

RSAC 2025 Conference

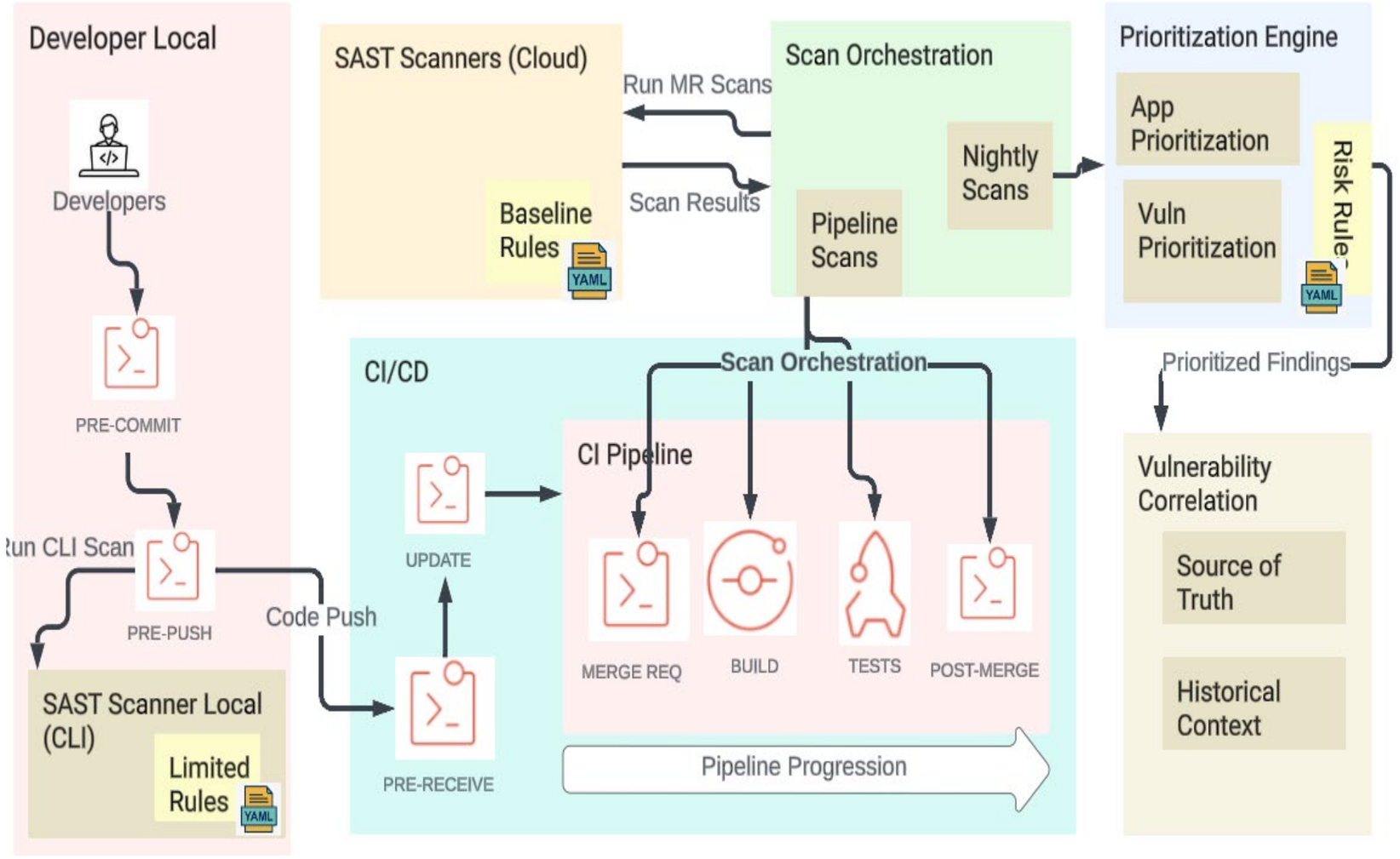
ST3 – Define Secure Build Pipeline

	Pre-Commit	Pre-Push	MR/PR	Post-Merge	Nightly
Purpose	Catch secrets, config issues	Code-level vulns before push	Validate vulns in code diff	Validate integrated code	Validate full repo
Implementation	Incremental Scan	Full Scan	Incremental Scan	Full Scan	Comprehensive scan
Enforcement	Instructional	Instructional	Block MR/PR	Risk-based Ticketing	Risk-based Ticketing

SoFi

RSAC 2025 Conference

ST4 - Develop Contextual Risk Engines



Prioritization

App Prioritization: factors in business criticality, data sensitivity, exposure

Vuln Prioritization: elevates findings that are reachable, exploitable, and high-impact

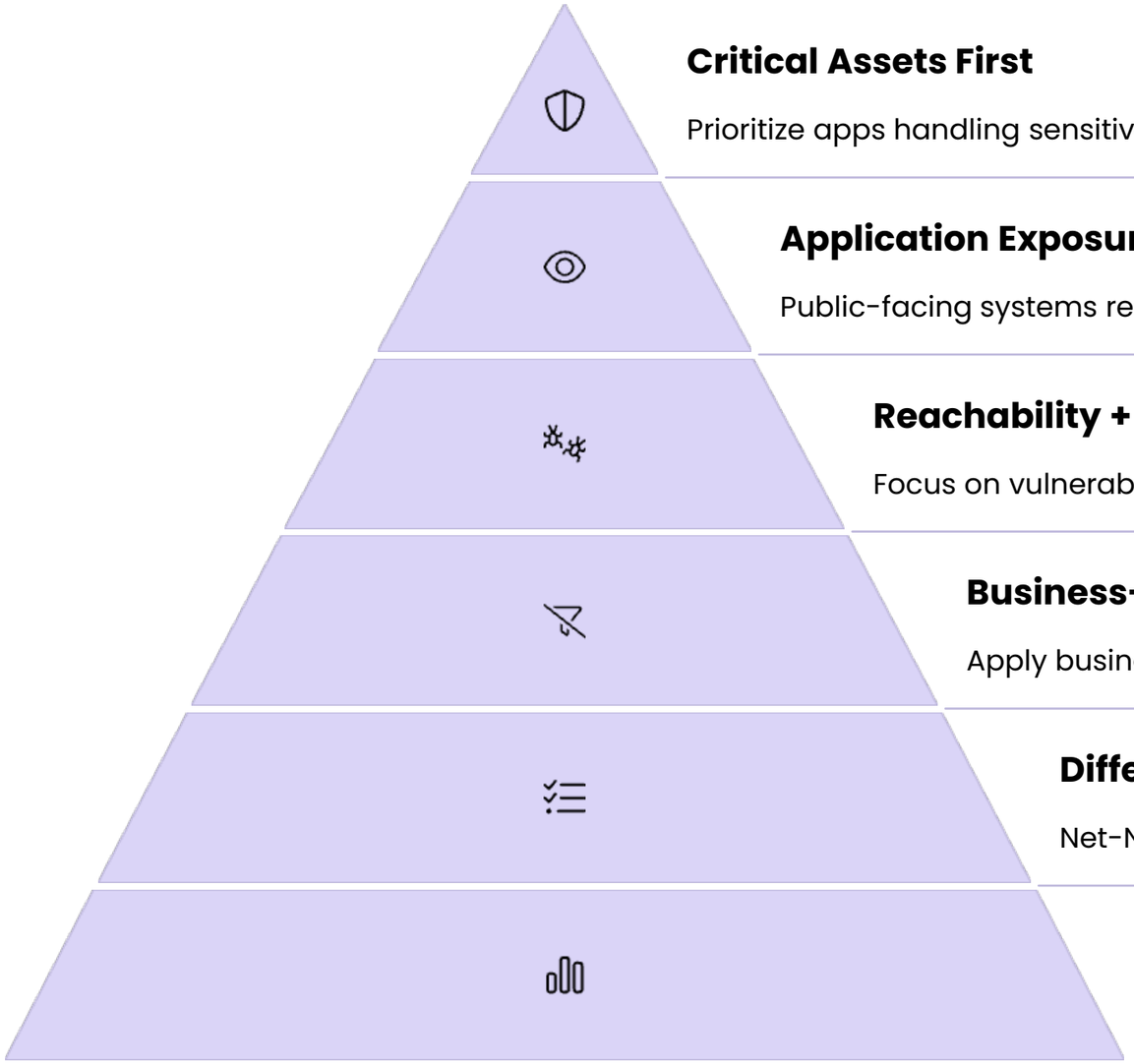
Controls Context: downgrades findings protected by strong compensating controls

Vuln Correlation

Consolidate findings across scanners

Correlate with historical findings

ST4 - Develop Contextual Risk Engines



Critical Assets First

Prioritize apps handling sensitive data or critical business functions.

Application Exposure

Public-facing systems receive higher priority

Reachability + Exploitability + Impact

Focus on vulnerabilities with high impact and ease of exploitation.

Business-Aware Filtering

Apply business/compliance context to reduce false positives and noise.

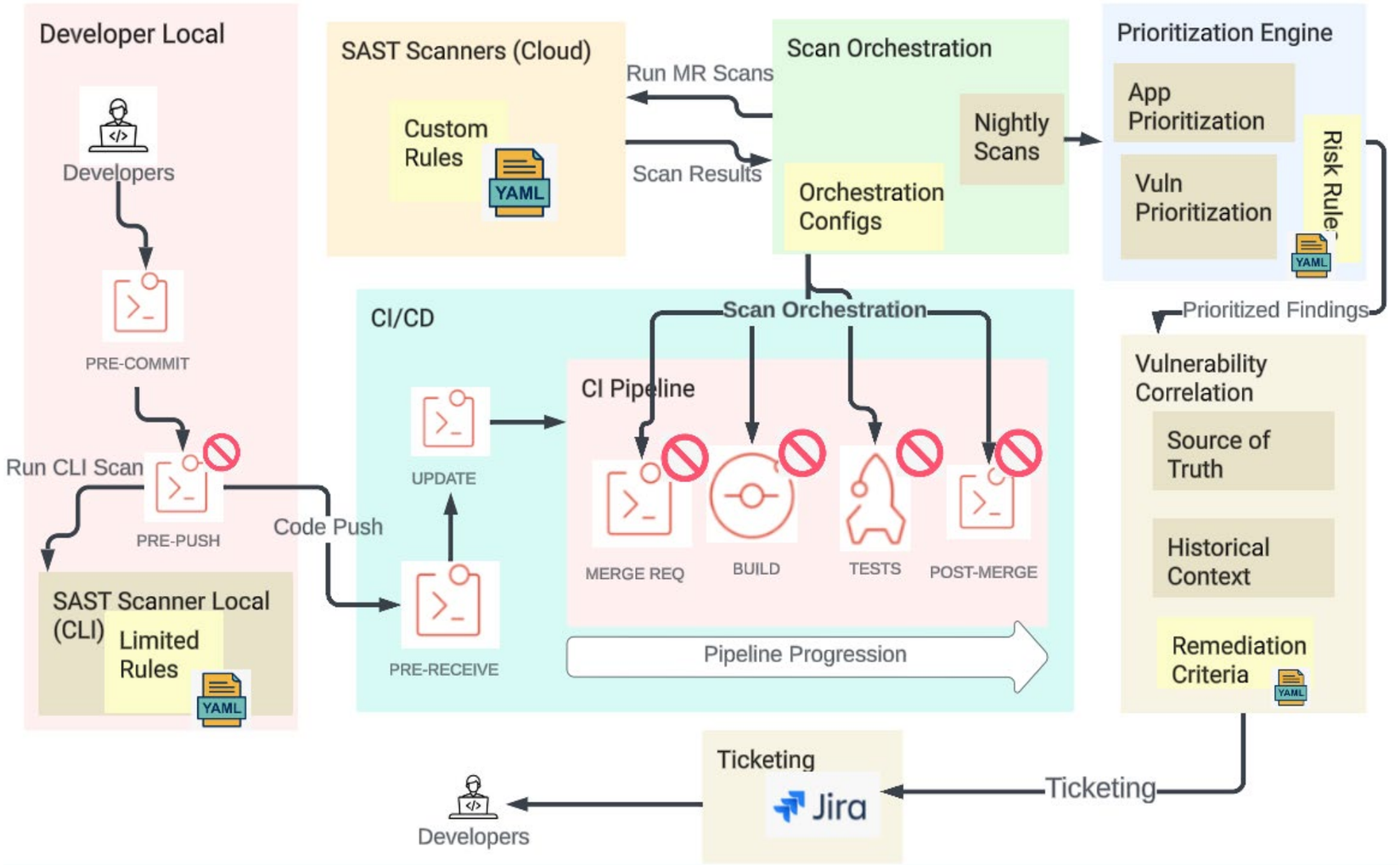
Differential Detection

Net-New MR Issues vs existing branch issues

Context Aggregation

Collate every piece of metadata to create a single source of truth

ST5 - Implement End to End Orchestration



Remediation Criteria

Define what gets fixed first - based on risk, efforts, resource availability, and cost benefits.

Set Blockers

Set blockers on critical pipeline gates to ensure net-new issues are identified and blocked

Issue Tickets with SLA

Post-Merge or pre-existing vulnerabilities are prioritized and ticketed.

Developer Notifications

Route findings via MR comments, Slack or IDE PopUps

Dev Feedback Forms

Allow Devs to mark as False Positive or Low Severity
Rate the clarity and actionability of the alert
Suggest rule improvements

SoFi

RSAC 2025 Conference

Break-Glass Scenarios

Authorized Bypass

Only select individuals with special roles can initiate break-glass procedures.

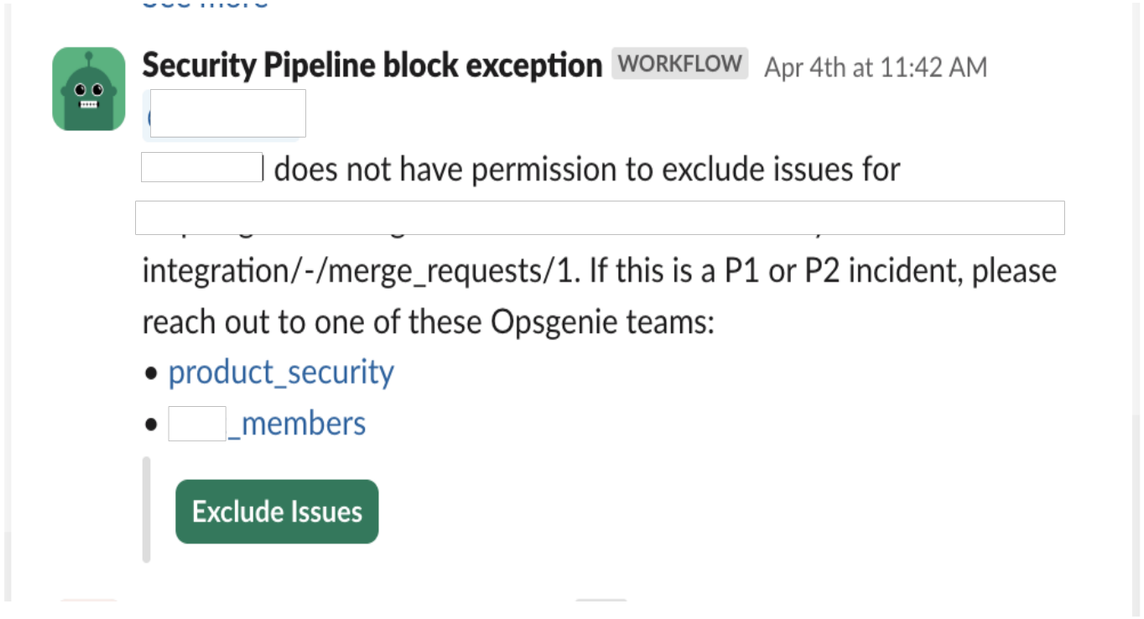
Time-Limited Exception

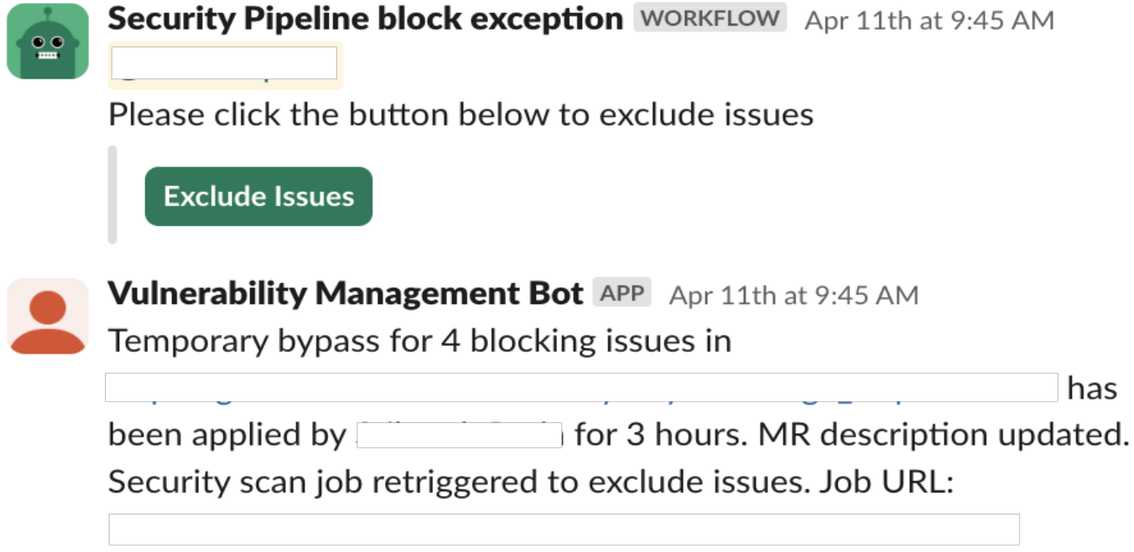
Temporary bypasses automatically expire after predefined periods. Prevents indefinite security gaps in the pipeline.

Automation Options

Self-service APIs and Slackbots streamline emergency bypass requests.

Enables quick response during P1 incidents.

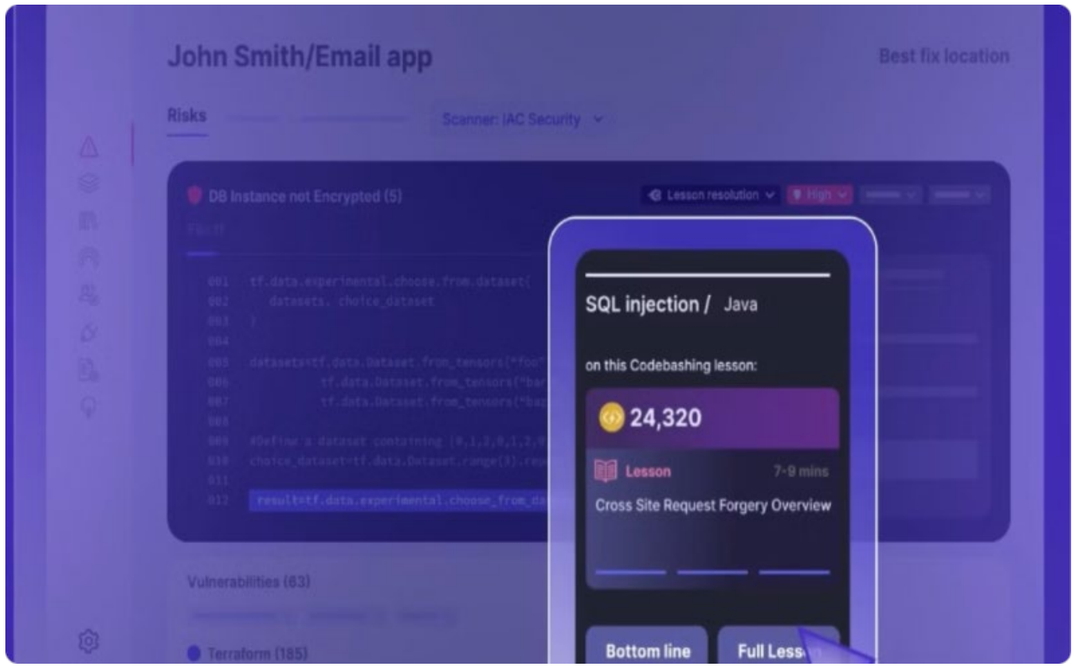

Security Pipeline block exception WORKFLOW Apr 4th at 11:42 AM
does not have permission to exclude issues for
integration/-/merge_requests/1. If this is a P1 or P2 incident, please reach out to one of these Opsgenie teams:
• product_security
• _members
Exclude Issues

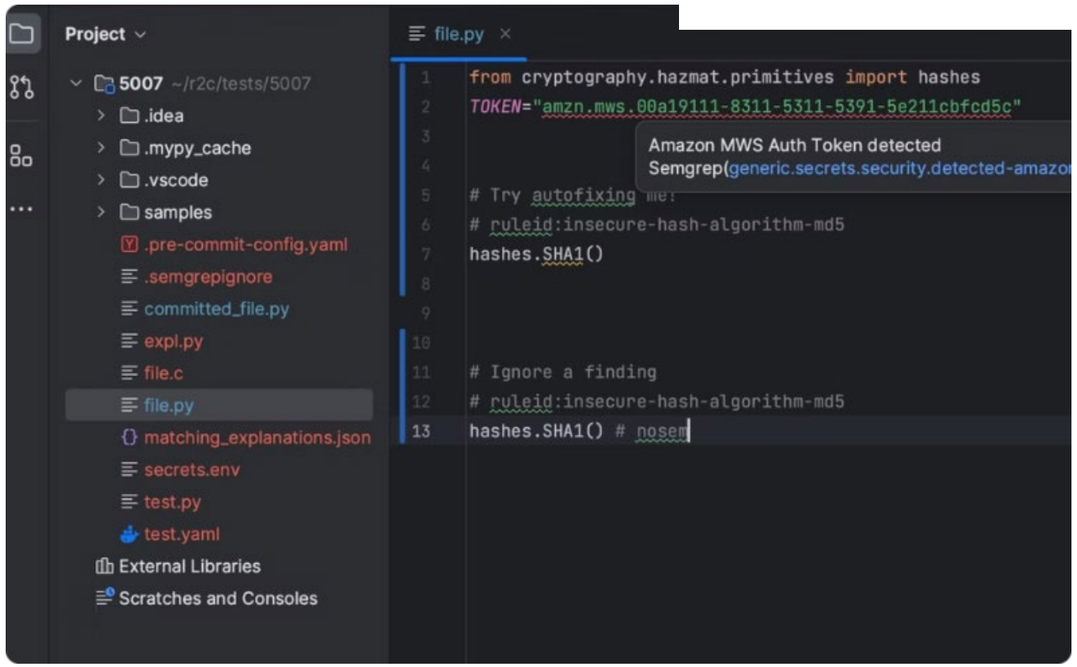

Security Pipeline block exception WORKFLOW Apr 11th at 9:45 AM
Please click the button below to exclude issues
Exclude Issues
Vulnerability Management Bot APP Apr 11th at 9:45 AM
Temporary bypass for 4 blocking issues in
has been applied by for 3 hours. MR description updated. Security scan job retriggered to exclude issues. Job URL:

SoFi

RSAC 2025 Conference

ST6 – Enhance Developer Adoption





Developer Training on Vulnerabilities

Experience: Interactive learning with real-time vulnerability insights

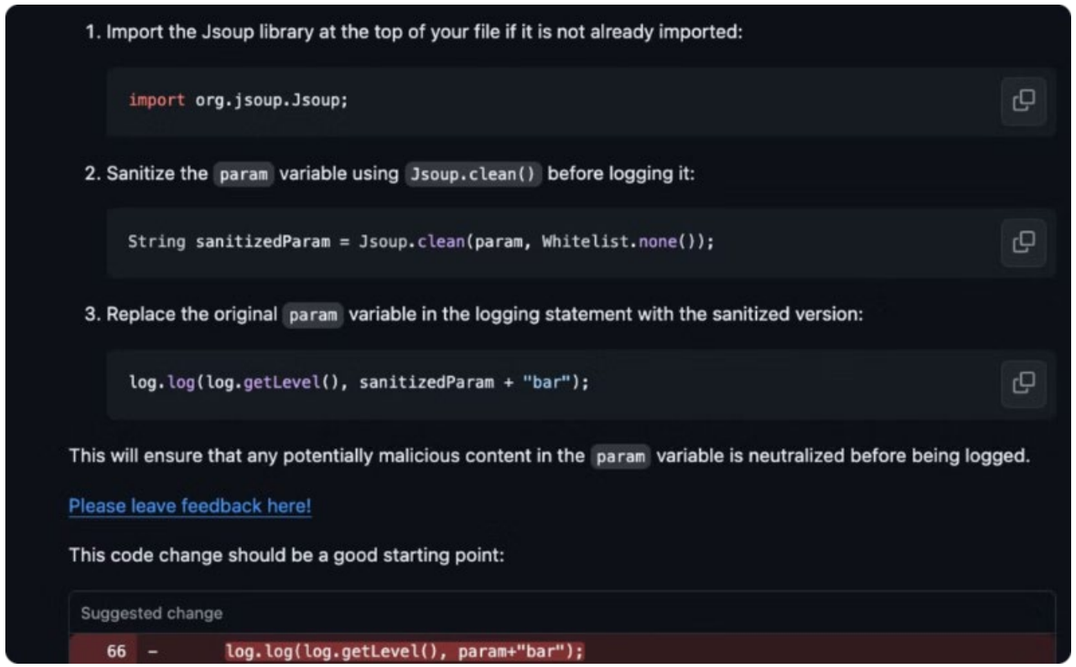
Rationale: Improves understanding while fixing vulnerabilities

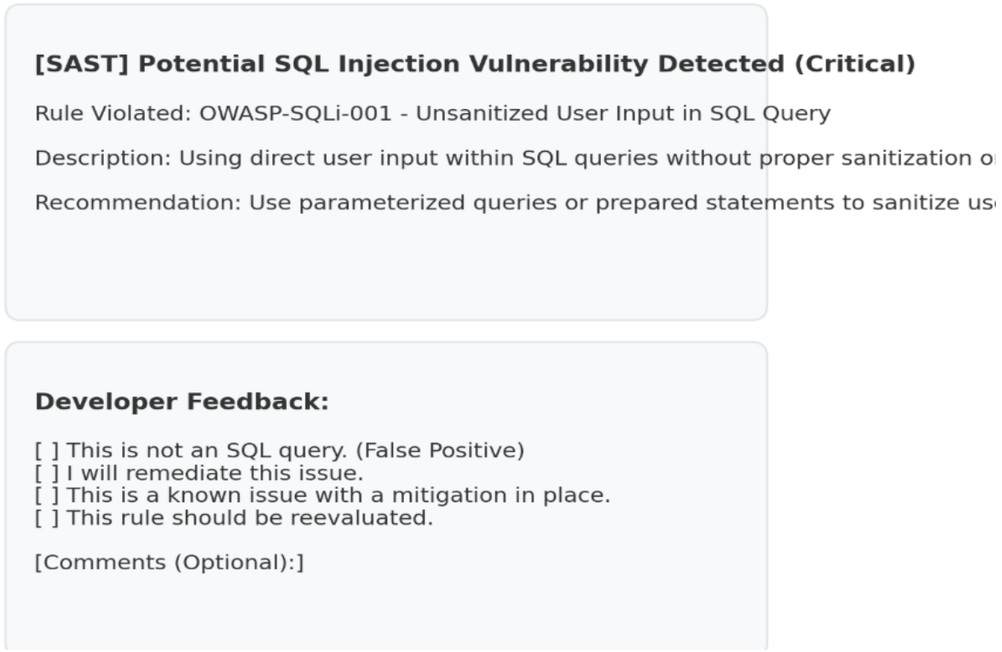
IDE Assist

Experience: Vulnerability insights directly in IDE during coding

Rationale: Enables Shift-Left security without enforcement

ST6 – Enhance Developer Adoption


1. Import the Jsoup library at the top of your file if it is not already imported:
import org.jsoup.Jsoup;
2. Sanitize the param variable using Jsoup.clean() before logging it:
String sanitizedParam = Jsoup.clean(param, Whitelist.none());
3. Replace the original param variable in the logging statement with the sanitized version:
log.log(log.getLevel(), sanitizedParam + "bar");
This will ensure that any potentially malicious content in the param variable is neutralized before being logged.
Please leave feedback here!
This code change should be a good starting point:
Suggested change
66 - log.log(log.getLevel(), param+"bar");


[SAST] Potential SQL Injection Vulnerability Detected (Critical)
Rule Violated: OWASP-SQLi-001 - Unsanitized User Input in SQL Query
Description: Using direct user input within SQL queries without proper sanitization o
Recommendation: Use parameterized queries or prepared statements to sanitize us
Developer Feedback:
[ ] This is not an SQL query. (False Positive)
[ ] I will remediate this issue.
[ ] This is a known issue with a mitigation in place.
[ ] This rule should be reevaluated.
[Comments (Optional):]

Smart Remediation Guidance

Experience: Contextual code comments with actionable recommendations in merge requests.

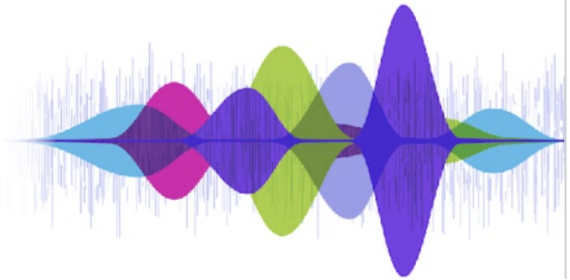
Rationale: Accelerates security issue resolution in pipelines.

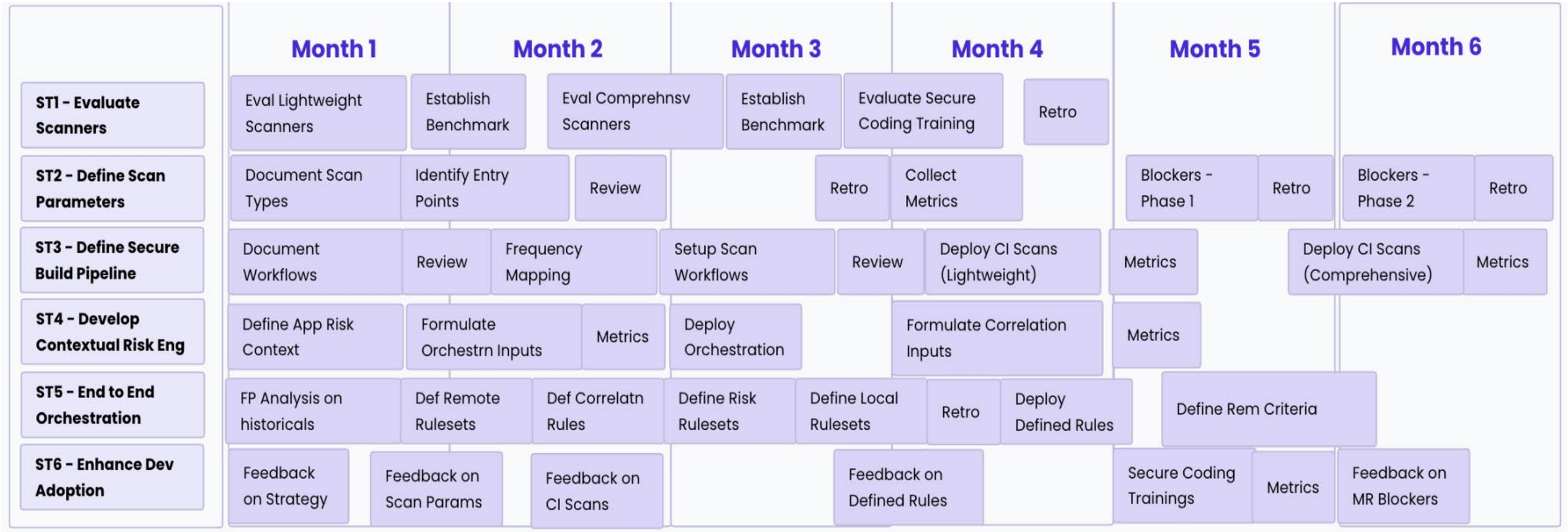
Continuous Feedback

Experience: Developers report policy conflicts or rule mismatches.

Rationale: Improves rules and reduces false positives through collaboration.

Strategy Roadmap





SoFi

RSAC 2025 Conference

Measure Success

What we achieved

3x

Faster Scan Speed

Distributed processing and pipeline tuning

60%

Faster Remediation

Streamlined workflows + actionable, high-signal findings

40%

Reduced Triage Fatigue

Contextual prioritization filtered out noise

30%

Fewer Redundant Alerts

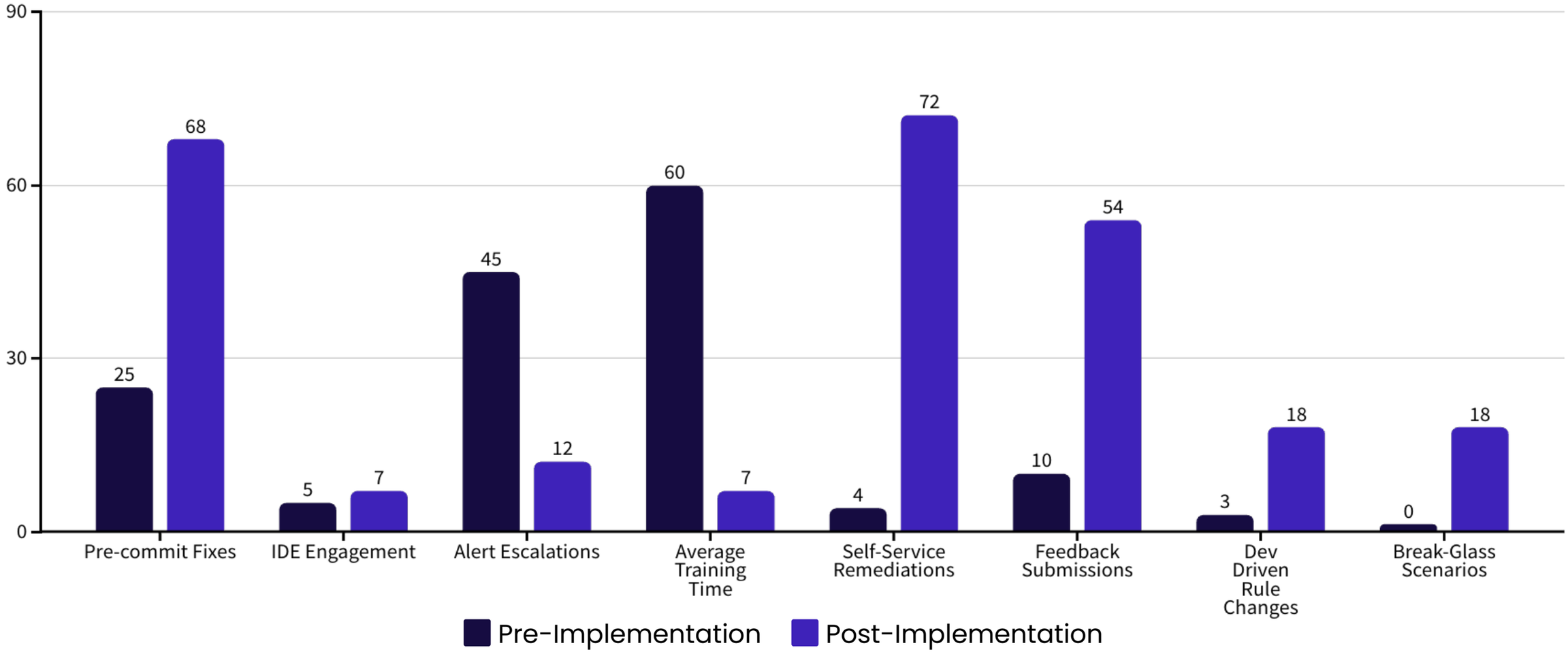
Smart correlation and historical context

100%

Tool Coverage without Disruption

Scan types aligned to risk, not one-size-fits-all enforcement

Developer Adoption & Engagement



SoFi

RSAC 2025 Conference

What's Next for SAST

Vibe Coding to 'Vibe Secure Coding'

Leverage AI not just to write code - but to write **secure code** with secure-by-default workflows

Real-time security recommendations during generation

AI-Augmented Rulesets

Transition from static rules to **ML-powered, context-aware analysis**.

Findings and rules evolve with developer feedback and real-world attack data

Prediction and Risk Modeling

SAST tools **predict where vulnerabilities are likely to occur** based on historical commit behavior and repo patterns.

Dynamically adjusts scan rules and depth based on real-time risk modeling

Shift-Right Feedback Rules

Feed real-world exploit attempts, bug bounty data, or runtime protection telemetry **back into SAST rule tuning**

Key Takeaways

1 Balance Technical + Human Factors

Tools + Dev Ex = Real Adoption

2 Context Matters for SAST

App & vuln context shape SAST impact

3 Pace your SAST scans - don't overwhelm the builds

Optimize performance by splitting fast MR scans from deeper nightly runs.

4 Use Dev Feedback to fine-tune everything

Embed a feedback loop to evolve rules and reduce noise over time.

5 **Focus on Broad Security Strategy beyond vulnerabilities**

Questions